# Better Be Lucky Than Good: Exceeding Expectations in MDP Evaluation

**Thomas Keller** and **Florian Geißer**
University of Freiburg
Freiburg, Germany
{tkeller,geisserf}@informatik.uni-freiburg.de

## Abstract

We introduce the MDP-Evaluation Stopping Problem, the optimization problem faced by participants of the International Probabilistic Planning Competition 2014 that focus on their own performance. It can be constructed as a meta-MDP where actions correspond to the application of a policy on a base-MDP, which is intractable in practice. Our theoretical analysis reveals that there are tractable special cases where the problem can be reduced to an optimal stopping problem. We derive approximate strategies of high quality by relaxing the general problem to an optimal stopping problem, and show both theoretically and experimentally that it not only pays off to pursue luck in the execution of the optimal policy, but that there are even cases where it is better to be lucky than good as the execution of a suboptimal base policy is part of an optimal strategy in the meta-MDP.

## Introduction

Markov Decision Processes (MDPs) offer a general framework to describe probabilistic planning problems of varying complexity. The development of algorithms that act successfully in MDPs is important to many AI applications. Since it is often impossible or intractable to evaluate MDP algorithms based on a theoretical analysis alone, the International Probabilistic Planning Competition (IPPC) was introduced to allow a comparison based on experimental evaluation. The idea is to approximate the quality of an MDP solver by performing a sequence of runs on a problem instance, and by using the average of the obtained results as an approximation of the expected reward. Following the optimal policy (i.e., the policy that maximizes the expected reward) leads to the best result in such a setting.

The work on this paper started with our preparation for IPPC 2014, where each solver had to perform *at least* 30 runs within a given time limit, while only *the last* 30 runs were used for evaluation. The decision when to stop the sequence of runs could be taken at any point of the evaluation with knowledge of the rewards that were collected in all previous runs. We describe the MDP-Evaluation Stopping Problem (MDP-ESP) as the optimization problem faced by IPPC participants that focus on their own performance, and show how it can be constructed as a meta-MDP with actions that correspond to the application of a policy on the base-MDP. Interestingly, the computation of the optimal policy is no longer the only objective of participating planners, and the fact that the execution of other policies on the base-MDP can be part of an optimal strategy for the MDP-ESP leads to a problem that is intractable in practice.

However, there are special cases where the MDP-ESP can be reduced to an instance of an optimal stopping problem (OSP). Two functions that depend only on the number of remaining runs – one that specifies the target reward that is necessary to stop, and one that gives the policy that is applied otherwise – suffice to describe an optimal policy. Based on these observations, we present four approximate algorithms for the general problem. A strategy that is inspired from a solution to the related Secretary Problem (SecP) (Dynkin 1963; Ferguson 1989) can be applied even when a policy for the base-MDP is computed online and not known in advance. Two other algorithms require the knowledge of the optimal policy and its expected reward. We show that the expected reward of the optimal policy is a lower bound for the expected performance of both strategies.

Our final algorithm switches between the application of the optimal policy and the policy with the highest possible outcome, which can be computed without notable overhead in the Trial-based Heuristic Tree Search (THTS) framework (Keller and Helmert 2013). We show theoretically and empirically that all algorithms outperform the naïve base approach that ignores the potential of optimizing evaluation runs in hindsight, and that it pays off to take suboptimal base policies in addition to the optimal one into account. Finally, we discuss the influence of the MDP-ESP on the results of IPPC 2014, and propose potential applications of our algorithms by discussing them in the context of related work.

## Background

**Markov Decision Processes.** In this paper we consider problems of planning and acting under uncertainty, where an agent interacts with an uncertain environment by performing a sequence of runs. The environment is described in terms of a finite-horizon MDP $M = \langle V, A, T, R, s_0, h \rangle$ (Puterman 1994; Bertsekas and Tsitsiklis 1996) with a factored representation of states (Boutilier, Dearden, and Goldszmidt 2000) that are induced by a set of state variables $V$ as $S = 2^V$. $A$ is a finite set of actions such that $A(s)$ gives the set of

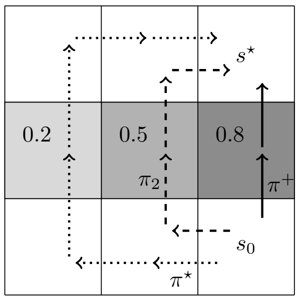

Figure 1: An example instance of the NAVIGATION domain with the policies $\pi^\star$ (dotted), $\pi_2$ (dashed) and $\pi^+$ (solid).

applicable actions in $s \in S$; $T : S \times A \times S \to [0, 1]$ is the transition function which defines the probability $\mathbb{P}[s' \mid s, a]$ that applying $a \in A(s)$ in $s \in S$ leads to $s' \in S$; $R(s, a)$ is the reward that is gained by applying $a \in A(s)$ in $s \in S$; $s_0$ is the initial state; and $h \in \mathbb{N}$ is the finite horizon.

The agent has access to the declarative model of the MDP (i. e., transition probabilities and reward function are known) to compute policies $\pi_1, \dots, \pi_n$ that are executed in a sequence of runs $(\phi_1^{\pi_1}, \dots, \phi_n^{\pi_n})$. Each policy $\pi$ in the set of policies $\Pi$ maps all states $s \in S$ to an action $a \in A(s)$. When a policy $\pi$ is applied in a state $s$, the current state of the environment transitions to successor state $s' \in S$ with probability $\mathbb{P}[s' \mid s, \pi(s)]$. A run is a sequence $\phi^\pi = (s_0, a_0, r_0 \dots, s_{h-1}, a_{h-1}, r_{h-1}, s_h)$ that starts in $s_0$ and where $h$ actions are applied according to $\pi$, i. e., the sequence is such that $a_t := \pi(s_t)$, $s_{t+1} \sim T(s_t, a_t, \cdot)$, and $r_t := R(s_t, a_t)$ for all $0 \le t < h$. We denote the accumulated reward of a run $\phi^\pi$ with $R(\phi^\pi) = \sum_{t=0}^{h-1} r_t$.

The expected reward of a policy $\pi$ can be computed as the solution to a set of equations that describes $V^\pi := V^\pi(s_0)$, which is given in terms of state-value functions $V^\pi(s)$ and action-value functions $Q^\pi(s, a)$, where

$$\begin{aligned} V^\pi(s) &= \begin{cases} 0 & \text{if } s \text{ is terminal} \\ Q^\pi(s, \pi(s)) & \text{otherwise, and} \end{cases} \\ Q^\pi(s, a) &= R(s, a) + \textstyle\sum_{s' \in S} \mathbb{P}[s' \mid s, a] \cdot V^\pi(s'). \end{aligned}$$

The optimal policy $\pi^\star$ can be derived from the related Bellman optimality equation (Bellman 1957; Bertsekas 1995) as the policy with the highest expected reward among all policies, i. e., as $\pi^\star := \arg\max_{\pi \in \Pi} V^\pi$.

Each policy $\pi$ induces a set of outcomes $\mathcal{O}^\pi$, which consists of each accumulated reward $r$ that can be achieved under application of $\pi$ paired with the probability of $r$, i. e., $\mathcal{O}^\pi = \{(r, \mathbb{P}[R(\phi^\pi) = r]) \mid \mathbb{P}[R(\phi^\pi) = r] > 0\}$. We call the highest possible outcome $P^\pi := \max_{(r,p) \in \mathcal{O}^\pi} r$ of a policy $\pi$ the potential of $\pi$. We abbreviate the policy with the highest potential among all policies with $\pi^+$. Moreover, we abbreviate the expected reward of $\pi^\star$ ($\pi^+$) with $V^\star$ ($V^+$), its potential with $P^\star$ ($P^+$), its set of outcomes with $\mathcal{O}^\star$ ($\mathcal{O}^+$) and a run under the policy with $\phi^\star$ ($\phi^+$).

An example MDP is depicted in Figure 1. It shows an instance of the NAVIGATION domain of IPPC 2011, where an agent is initially located in grid cell $s_0$ and aims to reach cell $s^\star$ by moving within the grid. On its way, the agent has to cross the middle row at some point, where it gets stuck with increasing probability from left (20%) to right (80%). The agent has no possibility to break free once it is stuck, and it receives a reward of $-1$ in each step unless it is located in $s^\star$. If we consider the IPPC horizon of $h = 40$, the agent receives an accumulated reward of $R(\phi^\star) = -6$, $R(\phi^{\pi_2}) = -4$, and $R(\phi^+) = -2$ if it successfully passes the middle row, and of $-40$ if it gets stuck regardless of the applied policy. The expected reward of $\pi^\star$ is $V^\star = -12.8$, and it induces the set of outcomes $\mathcal{O}^\star = \{(-6, 0.8), (-40, 0.2)\}$ with potential $P^\star = -6$. For $\pi^+$, we have $V^+ = -32.4$, $\mathcal{O}^+ = \{(-2, 0.2), (-40, 0.8)\}$, and $P^+ = -2$.

**The MDP-Evaluation Stopping Problem.** The problem we face in this paper is the MDP-ESP$_k^u$, where a sequence of at least $k > 0$ and at most $u \ge k$ runs is performed on an MDP. A strategy $\sigma$ assigns policies $\pi_1, \dots, \pi_n$ to runs on the MDP and stops the evaluation after $n$ ($k \le n \le u$) runs. The objective is to find a strategy $\sigma$ that maximizes the average accumulated reward of the last $k$ runs, i.e., where

$$\mathcal{R}_k^u(\sigma) := \frac{1}{k} \cdot \sum_{i=n-k+1}^{n} R(\phi_i^{\pi_i})$$

is maximal in expectation. The quality of a strategy $\sigma$ is measured in terms of its expected average reward $\mathbb{E}[\mathcal{R}_k^u(\sigma)]$.

An instance of the MDP-ESP not only optimizes the evaluation of a sequence of policies on a base-MDP, it can be described in terms of a meta-MDP itself. A state in the meta-MDP is given by a sequence of rewards $(r_1, \dots, r_n)$, where $r_i := R(\phi_i^{\pi_i})$ for $i = 1, \dots, n$ is the accumulated reward of the runs that were performed before reaching a state. The meta-MDP provides an action $a_\pi$ for each policy $\pi \in \Pi$, which encodes the execution of policy $\pi$ on the base-MDP. Furthermore, there is a single action $a_\otimes$ that encodes the decision to stop the MDP-ESP and evaluate the meta-run under $\sigma$ based on the result of the last $k$ runs on the base-MDP. We describe the transition function of the meta-MDP in terms of its actions: $a_\otimes$ is not applicable in a state $(r_1, \dots, r_n)$ if $n < k$, and it is the only applicable action in a state $(r_1, \dots, r_u)$. Its application leads deterministically to an absorbing terminal state and yields a reward $R((r_1, \dots, r_n), a_\otimes) = \frac{1}{k} \cdot \sum_{i=n-k+1}^{n} r_i$. The application of an action $a_\pi$ in state $(r_1, \dots, r_n)$ incurs no reward and leads to a state $(r_1, \dots, r_n, r)$, where $r$ is drawn according to the outcome function $r \sim \mathcal{O}^\pi$ of the executed policy $\pi$.

## Theoretical Analysis

**Upper and Lower Bounds.** Most optimization problems on MDPs have in common that the theoretical upper bound of the expected reward of a policy is less than or equal to the expected reward $V^\star$ of the optimal policy $\pi^\star$. Examples include the Multi-Armed Bandit problem and Online Reinforcement Learning (RL), where a logarithmic regret on $V^\star$ must be accepted (Lai and Robbins 1985) since all runs are evaluation runs and $\pi^\star$ must be derived from experience of the interaction with the environment. In Offline RL (or Probabilistic Planning), all runs are evaluation runs as well, but $\pi^\star$ can be computed before evaluation starts and $V^\star$ can hence be achieved if $\pi^\star$ is available (Sutton and Barto 1998).

This is different in the MDP-ESP. Since the agent is allowed to decide in hindsight if the last $k$ runs were good enough to be used for evaluation, there are strategies that allow an expected performance that is at least as good as $V^\star$ for all instances of the MDP-ESP$_k^u$. Moreover, it is impossible to achieve a result that is higher than $P^+$.

**Theorem 1.** *$V^\star \leq \max_\sigma \mathbb{E}[\mathcal{R}_k^u(\sigma)] \leq P^+$ for all $k > 0$ and $u \geq k$.*

**Proof sketch:** We start with a discussion of the lower bound $V^\star$ by considering the subset of instances where $u = k$. The MDP-ESP$_k^k$, where each performed run is an evaluation run reduces to Offline RL, and the optimal strategy is hence the strategy that only executes $\pi^\star$. (We denote the strategy that executes $\pi^\star$ in each run and never stops prematurely with $\sigma_{\pi^\star}$ in this proof sketch). Since the expected reward of each run under $\pi^\star$ is $V^\star$, the expected average reward of the whole sequence of $k$ runs is $V^\star$ as well. If we apply $\sigma_{\pi^\star}$ to instances where $u > k$, the additional, prepended runs have no effect as they are not used for evaluation. Therefore, $\mathbb{E}[\mathcal{R}_k^u(\sigma_{\pi^\star})] = V^\star$ for any instance of the MDP-ESP, and the lower bound is as stated in Theorem 1.

$P^+$ is an upper bound of the MDP-ESP$_k^u$ since it is the highest possible outcome of all policies, and it is therefore impossible to achieve a higher expected reward in a run and a higher expected average reward in a sequence of $k$ runs. Moreover, the bound is tight for the MDP-ESP$_k^\infty$, i. e., the subset of instances with an infinite number of runs and a finite number of evaluation runs. Since any sequence of outcomes will occur eventually in an infinite number of runs, the optimal strategy for the MDP-ESP$_k^\infty$ applies $\pi^+$ in every run until a sequence of $k$ runs in a row yields $P^+$, and the expected average reward of this strategy is $P^+$.

**Optimal Strategies.** Even though we have provided tight upper and lower bounds for the MDP-ESP, the expected reward of optimal policies in the space between the discussed extreme cases is not yet clear. It is not hard to show that the expected reward of the MDP-ESP$_k^u$ under an optimal strategy increases strictly from $V^\star$ to $P^+$ with increasing $u$ for all $k$ (unless $\pi^\star = \pi^+$ and $\pi^\star$ deterministic, in which case $\max_\sigma \mathbb{E}[\mathcal{R}_k^u(\sigma)] = V^\star = P^+$ for all $k > 0$ and $u \geq k$). We omit a formal proof sketch for space reasons, though, and turn our attention to the MDP-ESP$_1^u$ instead. It corresponds to a finite-horizon version of the House-Selling Problem (Karlin 1962), where offers come in sequentially for a house an agent wishes to sell. The offers are drawn from a known probability distribution, and the agent has to accept or decline each offer right after receiving it. The agent aims to sell the house for the highest price among at most $u$ offers. The subset of instances where only a single run is used for evaluation is interesting for our purposes because an optimal strategy can be described with two simple functions: the target reward function $t : \{1, \dots, u-k\} \to \mathbb{R}$ describes the average reward $t(n)$ of the last $k$ runs that must have been achieved in a state $(r_1, \dots, r_{u-n})$ to apply $a_\otimes$, and the policy application function app : $\{1, \dots, u-k\} \to \Pi$ specifies the policy that is taken otherwise.

| $n$ | $\pi^\star$ | $\pi_2$ | $\pi^+$ | app($n$) |
|---|---|---|---|---|
| 1 | $\mathbf{-12.8}$ | $-22$ | $-32.4$ | $\pi^\star$ |
| 2 | $\mathbf{-7.36}$ | $-8.4$ | $-10.64$ | $\pi^\star$ |
| 3 | $-6.272$ | $\mathbf{-5.68}$ | $-6.288$ | $\pi_2$ |
| 4 | $-5.936$ | $\mathbf{-4.84}$ | $-4.944$ | $\pi_2$ |
| 5 | $-5.768$ | $-4.42$ | $\mathbf{-4.272}$ | $\pi^+$ |
| 6 | $-5.654$ | $-4.136$ | $\mathbf{-3.8176}$ | $\pi^+$ |

Table 1: The optimal strategy for the MDP-ESP$_1^u$ on the NAVIGATION instance of Figure 1 applies app($n$) if the current result is less than $t(n)$ (in bold) and stops otherwise.

A solution for the MDP-ESP$_1^u$ is to compute these functions by applying backward induction, a popular method to solve full information optimal stopping problems where an increasing number of available runs $u$ is considered (Gilbert and Mosteller 1966). We know that it is optimal to apply $\pi^\star$ in the MDP-ESP$_1^1$, and the expected reward is $V^\star$, i. e., app$(1) = \pi^\star$. Now consider the MDP-ESP$_1^2$: if, after the first run, our current result is higher than $V^\star$, we stop the evaluation, since the remaining problem is exactly the MDP-ESP$_1^1$ with expected reward $V^\star$. Otherwise, we apply app$(1) = \pi^\star$. The target reward function is therefore such that $t(1) = V^\star$. The policy that is applied in the first run of the MDP-ESP$_1^2$, app$(2)$, can be computed as the policy that maximizes the expected reward given $t(1)$, which in turn allows the computation of $t(2)$ and so on.

Take for example the NAVIGATION domain that was presented earlier. We have app$(1) = \pi^\star$ and $t(1) = V^\star = -12.8$. If we apply $\pi^\star$ in the first run of the MDP-ESP$_1^2$, we achieve a reward of $-6$ with probability 0.8 and of $-40$ with probability 0.2. Since we prefer not to stop in the latter case, we get $t(2) = (0.8 \cdot (-6)) + (0.2 \cdot t(1)) = -7.36$. Table 1 shows these computations for all three policies of the NAVIGATION instance that are depicted in Figure 1. It reveals that it is optimal to execute $\pi^+$ if five or more runs are left, and to stop only if a run successfully crosses the middle row and yields a reward of $-2$. If three or four runs are left, the strategy proposes the execution of policy $\pi_2$, and $\pi^\star$ is executed only in the last two runs. The example shows that restricting to strategies that consider only $\pi^\star$ and $\pi^+$ is not sufficient for optimal behavior.

**Complexity.** It is not hard to see that finding an optimal strategy for the general MDP-ESP$_k^u$ is practically intractable. It corresponds to solving the meta-MDP with a search space of size $(|\Pi| \cdot \mathcal{O}_{\max})^u$ with $\mathcal{O}_{\max} = \max_{\pi \in \Pi} |\mathcal{O}^\pi|$, which is intractable even if $|\Pi|$ were manageable (which is usually not the case). We have discussed three special cases of the MDP-ESP, though, and we have shown that an optimal strategy for two of them – the MDP-ESP$_k^k$ and the MDP-ESP$_k^\infty$ – can be derived in constant time under the assumption that the cost of deriving policies in the base-MDP can be neglected. For the third, we have provided an algorithm that regards all outcomes of all policies $\pi \in \Pi$ in at most $(u-k)$ decisions, and it is hence linear in $u$, $|\Pi|$, and $\mathcal{O}_{\max}$. Even though the

| | $u = k$ | $k < u < \infty$ | $u = \infty$ |
|---|---|---|---|
| $k = 1$ | $O(1)$ | $O(u \cdot \lvert\Pi\rvert \cdot \mathcal{O}_{\max})$ | $O(1)$ |
| $1 < k < \infty$ | $O(1)$ | $O\big((\lvert\Pi\rvert \cdot \mathcal{O}_{\max})^u\big)$ | $O(1)$ |

Table 2: Complexity results for different instances of the MDP-ESP$^u_k$ given an oracle for the underlying base-MDP.

dependence on $|\Pi|$ is discouraging as the computation of all policies is intractable, it also shows that efficient approximations of good quality are possible if we consider only a subset of $\Pi$. The complexity results are summarized in Table 2. The manageable cases all have in common that two simple functions that map the number of remaining runs to a target reward and a policy suffice to describe the strategy. In the next section, we show how these ideas can be used to approximate the general case with strategies of high quality.

## Strategies for the MDP-ESP

We consider three possible states of a-priori information: first, we look at the case where $\pi^\star$ and $V^\star$ are unknown, and assume that the computation of a policy and its execution are interleaved. We continue with MDPs where $\pi^\star$ and $V^\star$ can be computed, and present two strategies, one that aims at avoiding bad luck and one that pushes its luck under execution of $\pi^\star$. In the last part of this section, we present a strategy that mixes $\pi^\star$ and $\pi^+$ and prove that it is theoretically superior to the other considered strategies.

**Secretary Problem.** While most instances of the IPPC are such that they cannot be solved in the given time, it is always possible to perform more than $k$ runs. Even if the available time is distributed equally among all planning steps beforehand, there are reasons for spare time: the PROST planner (Keller and Eyerich 2012) that is used for our experiments detects reward locks; it recognizes states with only one reasonable action; it is able to solve an encountered state even in larger MDPs if the remaining horizon is small; and it reuses decisions if it encounters a state more than once.

If the optimal policy is not available, the MDP-ESP$^u_k$ is similar to the SecP, which is a variant of the finite-horizon House-Selling Problem where the underlying probability distribution is not revealed to the agent. It involves a single secretarial position and $u$ applicants which are interviewed sequentially in a uniformly random order. Applicants can be ranked unambiguously, and the decision to hire a candidate has to be made right after the interview and is irrevocable. The objective is to have the highest probability of selecting the best applicant of the whole group, and it can be shown that an optimal solution is to reject the first $\lfloor \frac{u}{e} \rfloor$ applicants ($\approx 36.8\%$) and select the first subsequent candidate that is ranked higher than all candidates before (e.g., Ferguson 1989; Bruss 2000).

To apply the SecP strategy to the MDP-ESP$^u_k$, we pretend that all sequences of $k$ consecutive runs are independent, identically distributed data points. We perform $\lfloor \frac{u-k+1}{e} \rfloor$ runs and stop as soon as the last $k$ runs yield a higher average reward than all data points before. It is important to note that the data points are of course not independent and identically distributed in our setting – each data point depends on the previous one(s) unless $k = 1$, since two consecutive samples differ only in a single reward. Our empirical evaluation (where only $\pi^\star$ is executed) shows that the SecP is a strategy that improves over $V^\star$ significantly nonetheless.

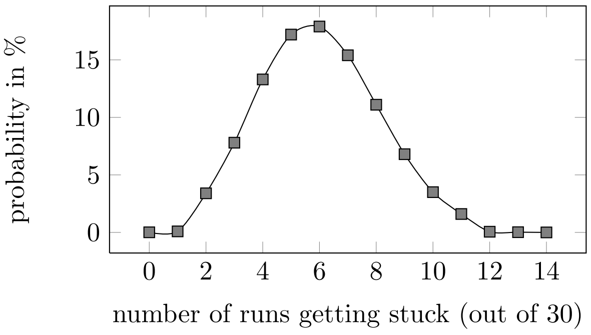

Figure 2: Probability of getting stuck $x$ times in 30 runs of the example NAVIGATION instance.

**Meet-The-Expectations.** The IPPC benchmarks offer MDPs of varying complexity, including some instances where $\pi^\star$ can be computed. However, it is always possible that the execution of a policy is unfortunate. Take, for example, the NAVIGATION instance from Figure 1. If we execute $\pi^\star$ for $k = 30$ runs, the expected reward $V^\star$ is achieved if the agent ends up stuck exactly six times. Figure 2, which depicts how likely it is that the agent gets stuck, reveals that the probability that it gets stuck more than six times is roughly 40%. A strategy that avoids bad luck if more than $k$ runs are available is a first step in the right direction. We call the strategy with $t_{\sigma_{\text{MTE}}}(n) = V^\star$ and $\text{app}_{\sigma_{\text{MTE}}}(n) = \pi^\star$ for all $n$ the Meet-The-Expectations (MTE) strategy.

**Theorem 2.** *$V^\star \leq \mathbb{E}[\mathcal{R}^u_k(\sigma_{\text{MTE}})] \leq P^\star$ for all $k > 0$ and $u \geq k$.*

**Proof sketch:** If $\pi^\star$ is deterministic, all inequalities are trivially equalities. Otherwise, both inequalities hold since only $\pi^\star$ is applied. The first is strict for $u > k$ since we accept lucky results and improve unlucky ones, and the second is strict even for most instances of the MDP-ESP$^\infty_1$ since MTE stops with a result between $V^\star$ and $P^+$.

**Pure Strategy.** We have presented a strategy that avoids bad luck while applying $\pi^\star$, so the question naturally arises how to push the envelope and aim for good luck. After all, Figure 2 shows that the probability of getting stuck less than six times is also approximately 40%. Since an optimal target reward function is intractable in practice even if $\text{app}_{\sigma_{\text{PS}}} = \pi^\star$ for all $n$, we use a simulation approach in the Pure Strategy (PS) to estimate $t_{\sigma_{\text{PS}}}$. PS performs a sequence of $m$ simulations $(\oslash_1, \dots, \oslash_m)$ ($m$ is a parameter of the algorithm), where each $\oslash_i$ consists of $u$ runs $(\phi^\star_{i1}, \dots, \phi^\star_{iu})$. We use the simulations to compute the target reward function as $t_{\sigma_{\text{PS}}}(n) = \text{median}(\mathcal{R}^n_{\max}(\oslash_1), \dots, \mathcal{R}^n_{\max}(\oslash_m))$, where $\mathcal{R}^n_{\max}(\oslash_i) = \max_{l \in \{1,\dots,n\}}(\frac{1}{k}\sum_{s=l}^{l+k-1} R(\phi^\star_{is}))$.

**Algorithm 1:** Mixed Strategy for MDP-ESP$^u_k$ with $u > k$

```
compute_mixed_strategy$(u, k, m)$:
    let all $t(n) \leftarrow -\infty$, app$(n) \leftarrow \pi^\star$ and $n_0 \leftarrow 1$
    for $i = 0, \dots, k$ do
        sample_run_sequences$(u, k, m, i)$
        update_strategy$(u, k, i)$
sample_run_sequences$(u, k, m, i)$:
    for $j = 1, \dots, m$ do
        for $n = 1, \dots, u$ do
            if $(n \bmod k) < i$ then $r_n \leftarrow$ sample$(\pi^+)$
            else $r_n \leftarrow$ sample$(\pi^\star)$
            if $n > k$ then
                $t_{ij}(n{-}k) \leftarrow \max_{l \in \{1,\dots,n\}} (\frac{1}{k} \sum_{s=l}^{l+k-1} r_s)$
    for $n = 1, \dots, u - k$ do
        $t_i(n) \leftarrow$ median$(t_{i1}(n), \dots, t_{im}(n))$
update_strategy$(u, k, i)$:
    for $n = u - k, \dots, n_0$ do
        if $t_i(n) > t(n)$ then $t(n) \leftarrow t_i(n)$
        else
            for $l = n_0, \dots, n$ do
                if $(l \bmod k) < i$ then app$(n) \leftarrow \pi^+$
            $n_0 \leftarrow n$ and return
```

**Theorem 3.** *$V^\star \leq \mathbb{E}[\mathcal{R}^u_k(\sigma_{\text{PS}})] \leq P^\star$ for all $k > 0$ and $u \geq k$. For all finite $k > 0$, $\mathbb{E}[\mathcal{R}^\infty_k(\sigma_{\text{PS}})] = P^\star$ and $\mathbb{E}[\mathcal{R}^u_k(\sigma_{\text{PS}})]$ is monotonically increasing in $u$ and converges to $P^\star$.*

**Proof sketch:** $V^\star$ and $P^\star$ are bounds since only $\pi^\star$ is applied and $\mathbb{E}[t_{\sigma_{\text{PS}}}(n)] \geq V^\star$ for a sufficiently large $m$. $\mathbb{E}[t_{\sigma_{\text{PS}}}(n)]$ increases monotonically in $n$ from $V^\star$ to a value $\leq P^\star$ for $u > k$ since the number of considered data points in the simulations grows, and it reaches $P^\star$ for $u = \infty$ and therefore $\mathbb{E}[\mathcal{R}^\infty_k(\sigma_{\text{PS}})] = P^\star$. $\mathbb{E}[\mathcal{R}^u_k(\sigma_{\text{PS}})]$ is monotonically increasing since the expected reward is bounded from below by a probability weighted sum of all target rewards.

**Mixed Strategy.** $\pi^+$ can not only be derived when $\mathcal{O}^\pi$ is available for all $\pi$, but it can be computed as $\pi^+ := \max_{\pi \in \Pi} P^\pi(s_0)$, which is described by the set of equations

$$P^\pi(s) = \begin{cases} 0 & \text{if } s \text{ is terminal} \\ W^\pi(s, \pi(s)) & \text{otherwise, and} \end{cases}$$
$$W^\pi(s,a) = R(s,a) + \max_{s' \in \text{succ}(s,a)} P^\pi(s'),$$

with succ$(s,a) := \{s' \mid \mathbb{P}[s' \mid s,a] > 0\}$. Note that the only difference to the Bellman optimality function is that $W^\pi(s,a)$ only cares about the best outcome while $Q^\pi(s,a)$ uses the weighted average of all outcomes. By turning these equations into assignment operators that extend the backup function of a THTS algorithm, we can describe algorithms – in our case a UCT$^\star$ variant – that derive $\pi^+$ as a side-effect of the $\pi^\star$ computation and without notable overhead.

While it is possible to use $\pi^+$ as the base-policy of PS, it turns out that $u$ has to be prohibitively large to outperform PS based on $\pi^\star$. Instead, we generate a policy that is inspired by our analysis of the MDP-ESP$^u_1$, where a function app$_{\sigma_{\text{MS}}}(n)$ is used to describe which policy is executed solely in terms of the number of remaining runs. We restrict ourselves to the policies $\pi^\star$ and $\pi^+$ in our version of a Mixed Strategy (MS), but adding more policies is an interesting (and certainly non-trivial) topic for future work. Initially, MS computes a function $t_0$ (the index stands for the number of runs under $\pi^+$ in each data point) which is equivalent to $t_{\sigma_{\text{PS}}}$. MS, which is depicted in Algorithm 1, continues by performing simulations where $\pi^+$ is executed in $i$ out of $k$ runs. The functions $t_{\sigma_{\text{MS}}}$ and app$_{\sigma_{\text{MS}}}$ are updated after the $i$th iteration by finding the largest $n$ where $t(n) \geq t_i(n)$, i. e., by finding the element in the sequence where the number of runs is small enough that an additional execution of $\pi^+$ does not pay off anymore. Note that our implementation stops the computation prematurely when a $t_i$ does not alter $t_{\sigma_{\text{MS}}}$ in the update procedure (unlike the depicted Algorithm 1).

**Theorem 4.** *$\mathbb{E}[\mathcal{R}^u_k(\sigma_{\text{PS}})] \leq \mathbb{E}[\mathcal{R}^u_k(\sigma_{\text{MS}})]$ for all $k > 0$ and $u \geq k$ and $\mathbb{E}[\mathcal{R}^\infty_k(\sigma_{\text{MS}})] = P^+$ for all finite $k > 0$.*

**Proof sketch:** If we assume that the number of simulations is sufficiently high, then it is either not beneficial to apply $\pi^+$ and MS reduces to PS, or it is beneficial and $\mathbb{E}[\mathcal{R}^u_k(\sigma_{\text{MS}})] > \mathbb{E}[\mathcal{R}^u_k(\sigma_{\text{PS}})]$. MS converges towards $P^+$ with an increasing number of $u$ since at some point it pays off to only apply $\pi^+$ with an expected reward of $P^+$ in the limit.

## Experimental Evaluation

To evaluate our algorithms empirically, we perform experiments on the domains of IPPC 2011 and 2014. We use the UCT$^\star$ algorithm (Keller and Helmert 2013) to solve the base-MDP, an algorithm that is specifically designed to find high-quality policies in finite-horizon MDPs even of larger size when a declarative model of the MDP is provided. It is a THTS algorithm that can be described in terms of four ingredients: the action selection is based on the UCB1 formula (Auer, Cesa-Bianchi, and Fischer 2002), Monte-Carlo sampling is used to simulate stochastic outcomes, and a partial Bellman backup function is used to propagate collected information in the search tree. Since we use the implementation of UCT$^\star$ that comes with the probabilistic planning system PROST, the used heuristic function is the default heuristic of the planner. It performs a lookahead based on a sequence of iterative deepening searches on the most-likely determinization of the MDP (Keller and Eyerich 2012).

We have altered the THTS framework to perform a sequence of searches with an increasing horizon, a change that is inspired by the Reverse Iterative Deepening approach that is used in GLUTTON (Kolobov et al. 2012). A higher number of instances can be solved since state-values of solved states are reused, which occurs more often if the horizon is increased iteratively (the possibly weaker anytime performance is not important here). The resulting algorithm is able to solve 34 instances of the 120 existing IPPC benchmarks: four of CROSSING TRAFFIC, five of ELEVATORS, three of GAME OF LIFE, all NAVIGATION instances, and six both of SKILL TEACHING and TRIANGLE TIREWORLD. Apart from the ELEVATORS domain, where $\pi^\star$ can be derived for the instances 1, 2, 4, 7, and 10, the instances with the lowest indices are solved. The number of evaluation runs $k$ is set to 30 in all experiments, which corresponds to the number of

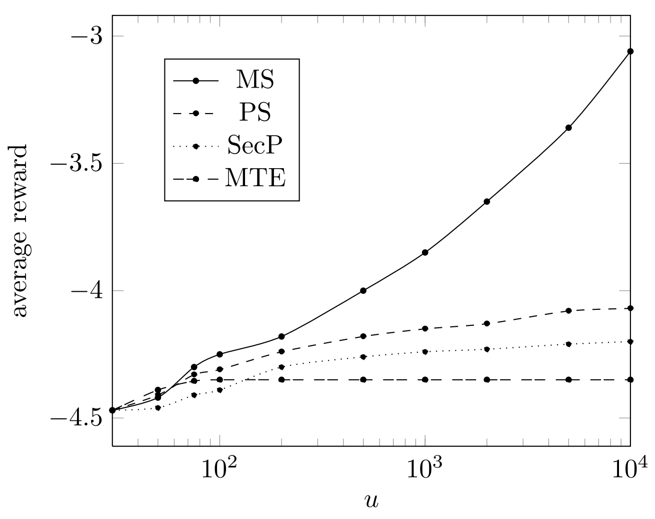

Figure 3: Results for an increasing number of available runs $u$ on the first instance of the CROSSING TRAFFIC domain with $V^\star \approx -4.43$, $P^\star = -4$, and $P^+ = -2$.

evaluation runs at both IPPC 2011 and 2014, and the values for $u$ are increased from 30 to 10000. Each experiment is conducted 20 times and average results are reported.

Figure 3 shows the average reward of the experiments on the first instance of CROSSING TRAFFIC with increasing $u$. We have selected the instance since comparably small values of $u$ showcase our theoretical results. Nevertheless, if $u$ is large enough, any instance could have been selected. Table 3 shows normalized IPPC scores which are computed over the average of the results of the experiment sets for $u = 200$ and $u = 1000$. A score of 1.0 is assigned to the best performing strategy, and a score of 0.0 is assigned to an artificial baseline solver with an average reward of $V^\star$ in all instances. All other scores are normalized with the procedure that is used at IPPC with the described minimal and maximal values.

The expected reward in the depicted CROSSING TRAFFIC instance is $V^\star \approx -4.43$. The simple MTE strategy is already an improvement over the baseline solver. It reliably avoids bad luck already with only a few extra runs. However, it has converged to $-4.35$, a value that is only little above $V^\star$, in the CROSSING TRAFFIC instance already with $u = 100$ and does not improve any further. The same can be observed in Table 3, where the result does not improve when $u$ is increased from 200 to 1000. In that experiment set, merely 35 to 40 runs suffice to avoid bad luck reliably over all instances, and in between 100 and 200 runs suffice for convergence. Except for special cases like an MDP-ESP$^u_1$ with $|\{(r,p) \in \mathcal{O}^\star \mid r \geq V^\star\}| = 1$, MTE converges to a value that is greater than $V^\star$ yet less than $P^\star$.

The SecP strategy does not suffer from this problem – the larger $u$, the better the result in our experiments. It quickly outperforms MTE even though the availability of the optimal policy is no condition for the applicability of the strategy. It should nevertheless be noted that the presented SecP results are based on an implementation that executes the optimal policy in all experiments to allow a better comparison of the strategies. In this setting, the SecP converges to $P^\star$ with growing $u$: since only $\pi^\star$ is applied, it cannot improve over $P^\star$, and since $\lfloor \frac{u-k+1}{e} \rfloor$ grows with $u$, the target reward and in turn the expected average reward approach $P^\star$.

The two sampling-based strategies yield comparable results in the experiment on all solvable IPPC instances that is given in Table 3, and both outperform the other considered algorithms significantly and in all domains. Obviously, simulation based approaches are well-suited to create strategies of high quality. It is not surprising that PS outperforms MTE with increasing $u$ since PS converges to $P^\star$ according to Theorem 3 while MTE usually does not, and since PS reduces to MTE if it ever were reasonable. The theoretical relation between the performance of PS and SecP is an open question, but it appears that the latter converges to $P^\star$ with a slower pace. PS often has an edge over MS when $u$ and $k$ are close, since applying $\pi^+$ is rarely reasonable in these cases and MS can hence only be misled by its additional possibilities. Increasing the number of simulations $m$ will neglect the slight advantage PS has in some instances. The larger $u$ compared to $k$, the larger the advantage of MS over PS. Figure 3, which depicts one of the smaller instances of the used benchmarks, shows this clearly: MS quickly outperforms all other strategies (as soon as it starts to mix in runs under $\pi^+$) and converges to $P^+ = -2$. Table 3 also supports this claim since MS outperforms PS in all domains with $u = 1000$. The only exception is the ELEVATORS domain, where $P^\pi(s_0) = 0$ for all policies $\pi$ since there is a small chance that no passenger shows up. If ties were broken in favor of better action-values in our implementation of $\pi^+$ (instead of uniformly at random), MS and PS would perform equally in the ELEVATORS domain.

## Discussion

We started with the work at hand due to the evaluation schema that was used at IPPC 2014. Only three IPPC solvers made use of the rule that more than 30 runs are allowed: both versions of PROST and G-PACK, a variant of the GOURMAND planner (Kolobov, Mausam, and Weld 2012). The latter does not reason over the MDP-ESP, though. It simplifies the original MDP by considering at most $N$ outcomes for all actions, and computes and executes the policy with highest expected reward in the simplified MDP. If time allows, this process is repeated with a larger $N$, which is the only reason that more than $k$ evaluation runs are performed.

Therefore, only our submissions actually considered the MDP-ESP as the relevant optimization problem. However, most of the work described in this paper was done *after* the competition – only the SecP strategy and a PS variant with a target reward that is independent from the number of remaining runs were applied at IPPC 2014. Note that both are strategies that aim at optimizing the evaluation of $\pi^\star$ or a near-optimal policy. PROST 2011 applied the SecP strategy in 33 out of 80 instances, while PROST 2014 used it in 28 and PS in another six instances. Even though we were able to improve the average reward in 22 (19) instances with the SecP strategy and in five with the PS variant in the 2014 (2011) version, the total IPPC scores are mostly unaffected: had we stopped evaluation after 30 runs in all instances with both

| | CROSSING | | ELEVATORS | | GAME | | NAVIGATION | | SKILL | | TIREWORLD | | Total | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $u$ | 200 | 1000 | 200 | 1000 | 200 | 1000 | 200 | 1000 | 200 | 1000 | 200 | 1000 | 200 | 1000 |
| **MTE** | 0.21 | 0.21 | 0.26 | 0.26 | 0.28 | 0.28 | 0.37 | 0.37 | 0.23 | 0.23 | 0.27 | 0.27 | 0.27 | 0.27 |
| **SecP** | 0.34 | 0.45 | 0.34 | 0.59 | 0.27 | 0.52 | 0.57 | 0.85 | 0.37 | 0.63 | 0.38 | 0.68 | 0.38 | 0.62 |
| **Pure** | **0.59** | 0.84 | **0.59** | **0.98** | 0.63 | **0.98** | **0.74** | 0.93 | 0.57 | **0.97** | **0.62** | 0.96 | 0.62 | 0.94 |
| **Mixed** | 0.57 | **1.00** | 0.55 | 0.93 | **0.68** | **0.98** | 0.73 | **1.00** | **0.60** | 0.96 | **0.62** | **0.99** | **0.63** | **0.98** |

Table 3: IPPC scores of the proposed algorithms for the MDP-ESP$^u_{30}$ on instances of IPPC 2011 and 2014 that can be solved with PROST. The expected reward $V^\star$ of the optimal policy $\pi^\star$ is used as the minimum for normalization.

solvers, the final result would differ only slightly with total IPPC scores of 0.816 (-0.009) and 0.773 (+0.004) for the PROST versions and of 0.739 (+0.005) for G-PACK. Even though the differences are small (we believe this is due to the poor minimum policy at IPPC, the individual results indicate that considering the MDP-ESP does pay off), we would like to emphasize that the PROST competition results should not be used for comparison. Otherwise, this includes instances where PROST achieved a result that is far above $V^\star$.

There are many applications for OSPs, including sponsored search (e.g., Babaioff et al. 2007; Zhou and Naroditskiy 2008), online auctions (e.g., Hajiaghayi, Kleinberg, and Parkes 2004), or optimal stock market behavior (e.g., Griffeath and Snell 1974; Shiryaev, Xu, and Zhou 2008). Most applications are based on variants of the SecP that differ in the number of applicants that must be selected as in the Multiple-Choice SecP (MCSP) (Freeman 1983), that have full information as in the House-Selling Problem or where the selected values must be maximized under constraints on each element as in the Online Knapsack Problem (Marchetti-Spaccamela and Vercellis 1995). The MDP-ESP differs from the MCSP in two details: first, the selected applicants must show up consecutively, and second, the probability distribution that gives the next sample must be selected from a known set of probability distributions.

The former difference does not alter the applicability of our algorithms, since scenarios where $k$ consecutive data points must be selected exist, e. g., with resources that decay with time, goods on an assembly line, or in traffic control. Moreover, our algorithms can also be applied to a variant of the MDP-ESP where $k$ results can be selected in arbitrary order. An exemplary application is, as in the MCSP, the plan to hire $k$ employees. In our scenario, all applicants have provided application documents, which allow the estimation of a probability distribution over the candidate's aptitude (e. g., grades or experience influence the expectation and the rest of the CV the variance and hence the potential). We invite at most $u$ of the applicants (more than $u$ applicants are necessary since we do not restrict the number of times a "type of applicant" is selected), and we have to decide right after the interview if we hire the candidate with knowledge of the aptitude. This problem differs from the MDP-ESP only in the way the $k$ applicants are selected. However, it is easy to see that Theorem 1 also holds, that the MTE algorithm can be applied with the same bounds on the expected reward (i. e., Theorem 2 holds) and that Theorems 3 and 4 hold for versions of PS and MS where line 12 of Algorithm 1 is replaced with an equation that sums the $k$ largest rewards independently from their position. Since this is a simple and useful generalization of the popular MCSP, it is well-suited to describe existing OSP applications more realistically.

## Conclusion

We have shown how the MDP-ESP is constructed as a meta-MDP where actions encode the execution of a policy in an underlying base-MDP. The expected reward of the optimal policy of the base-MDP is a lower bound for optimal strategies in the MDP-ESP. We have derived a procedure from the Bellman optimality equation to compute the policy that maximizes its potential, and have presented an upper bound for MDP-ESP strategies that corresponds to the potential of $\pi^+$. While the general MDP-ESP is intractable in practice, we have shown that there are special cases – the MDP-ESP$^k_k$, the MDP-ESP$^\infty_k$, and the MDP-ESP$^u_1$ – where the knowledge of $\pi^\star$ or $\pi^+$ suffices to compute an optimal strategy.

We have introduced four different strategies for the MDP-ESP based on our theoretical analysis. A strategy that is derived from the related SecP not only allows us to treat MDPs as MDP-ESP instances even though the optimal strategy cannot be computed , but is furthermore a strategy of high quality that allows to exceed the average expected reward of the underlying policy both in the experiments presented in this paper and at IPPC 2014. If $\pi^\star$ is available, we show that avoiding bad luck is already an improvement over a base policy that stops after $k$ runs of $\pi^\star$. However, by pushing the luck under application of the optimal policy, we derive a strategy that converges towards $P^\star$ and hence to a result that can only be achieved in a very lucky set of evaluation runs. We showed empirically that the corresponding strategy PS is of high quality if $u$ and $k$ are similar. The use of MS, which switches between executing $\pi^\star$ and $\pi^+$, becomes more appealing with an increasing difference between $u$ and $k$. It outperforms all other approaches significantly in our empirical evaluation and demonstrates that it is indeed sometimes better to be lucky than good.

**Acknowledgments.** We would like to thank Scott Sanner from ANU Canberra and Malte Helmert from the University of Basel for pointing us to the Secretary Problem, and Andrey Kolobov from Microsoft Research, Redmond for valuable information about GLUTTON and GOURMAND.

## References

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Journal of Machine Learning Research* 47:235–256.

Babaioff, M.; Immorlica, N.; Kempe, D.; and Kleinberg, R. 2007. A Knapsack Secretary Problem with Applications. In *Proceedings of the 10th International Workshop on Approximation (APPROX 2007)*, 16–28. Berlin, Heidelberg: Springer-Verlag.

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press.

Bertsekas, D., and Tsitsiklis, J. 1996. *Neuro-Dynamic Programming*. Athena Scientific.

Bertsekas, D. 1995. *Dynamic Programming and Optimal Control*. Athena Scientific.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2000. Stochastic Dynamic Programming with Factored Representations. *Artificial Intelligence (AIJ)* 121(1–2):49–107.

Bruss, F. T. 2000. Sum the Odds to One and Stop. *The Annals of Probability* 28(3):1384–1391.

Dynkin, E. B. 1963. The Optimum Choice of the Instant for Stopping a Markov Process. *Soviet Mathematics Doklady* 4.

Ferguson, T. S. 1989. Who Solved the Secretary Problem? *Statistical Science* 4(3):282–289.

Freeman, P. R. 1983. The Secretary Problem and its Extensions: A Review. *International Statistical Review* 51(2):189–206.

Gilbert, J. P., and Mosteller, F. 1966. Recognizing the Maximum of a Sequence. *Journal of the American Statistical Association* 61(313):35–73.

Griffeath, D., and Snell, J. L. 1974. Optimal Stopping in the Stock Market. *The Annals of Probability* 2(1):1–13.

Hajiaghayi, M. T.; Kleinberg, R.; and Parkes, D. C. 2004. Adaptive Limited-supply Online Auctions. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, 71–80. New York, NY, USA: ACM.

Karlin, S. 1962. Stochastic Models and Optimal Policy for Selling an Asset. *Studies in Applied Probability and Management Science* 148–158.

Keller, T., and Eyerich, P. 2012. PROST: Probabilistic Planning Based on UCT. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 119–127. AAAI Press.

Keller, T., and Helmert, M. 2013. Trial-based Heuristic Tree Search for Finite Horizon MDPs. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 101–105.

Kolobov, A.; Dai, P.; Mausam; and Weld, D. 2012. Reverse Iterative Deepening for Finite-Horizon MDPs with Large Branching Factors. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 146–154.

Kolobov, A.; Mausam; and Weld, D. 2012. LRTDP vs. UCT for Online Probabilistic Planning. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)*, 1786–1792.

Lai, T. L., and Robbins, H. 1985. Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics* 6(1):4–22.

Marchetti-Spaccamela, A., and Vercellis, C. 1995. Stochastic On-line Knapsack Problems. *Mathematical Programming* 68(1):73–104.

Puterman, M. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

Shiryaev, A.; Xu, Z.; and Zhou, X. Y. 2008. Thou Shalt Buy and Hold. *Quantitative Finance* 8(8):765–776.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press.

Zhou, Y., and Naroditskiy, V. 2008. An Algorithm for Stochastic Multiple-Choice Knapsack Problem and Keywords Bidding. In *Proceedings of the 17th International World Wide Web Conference (WWW 2008)*, 1175–1176.